Towards a System for Dynamic Language Resources in LOD

Kiril Simov
Linguistic Modelling Department
IICT-BAS, Sofia, Bulgaria
kivs@bultreebank.org

Abstract

Formalization and representation of the language resources life cycle in a formal language to support the creation, update and application of the language resource instances is made possible via the developments in the area of ontologies and Linked Open Data. In the paper we present some of the basic functionalities of a system to support dynamic language resources.

1 Introduction

Recent developments in Natural Language Processing (NLP) are heading towards knowledge rich resources and technology. Integration of linguistically sound grammars, sophisticated machine learning settings and world knowledge background is possible given the availability of the appropriate resources: deep multilingual treebanks, which represent detailed syntactic and semantic information. Such knowledge is mainly present in deeply processed language resources like HPSG-based (LFG-based) treebanks (RedWoods treebank, DeepBank, and others). The inherent characteristics of these language resources is their dynamic nature. They are constructed simultaneously with the development of a deep grammar in the corresponding linguistic formalism. The grammar is used to produce all potential analyses of the sentences within the treebank. The correct analyses are selected manually on the base of linguistic discriminators which would determine the correct linguistic production. The annotation process of the sentences provides feedback for the grammar writer to update the grammar. The life cycle of a dynamic language resource can be naturally supported by the semantic technology behind the ontology and LOD - modeling the grammatical knowledge as well as the annotation knowledge; supporting the annotation process; reclassification after changes within the grammar; querying the available resources; exploitation in real applications. The addition of a LOD component to the system would facilitate the exchange of language resources created in this way and would support the access to the existing resources on the web.

The structure of this paper is as follows: next section introduces related work; Section 3 discusses the main requirements of Knowledge-based Language Resources; Section 4 presents dynamic treebanking as an example of a Knowledge-based Language Resource; Section 5 concludes the paper.

2 Related work

Linked Open Data has been in active development during the last decade. (Bizer et al., 2009) defines the principles for publishing RDF data on the Web:

- Use URIs as (unique) names for things,
- Use HTTP URIs so that people can look up those names,
- When someone looks up a URI, provide useful information, using Web standards such as RDF, and SPARQL,
- Include links to other URIs, so that they can discover more things.

Usually LOD is grouped in datasets, equipped with ontology. Examples of LOD Datasets are: (1) **DBPedia.** DBPedia dataset is based on extraction of structural information from Wikipedia, which is presented in RDF form; (2) **Freebase.** Freebase is a community-curated database of well-known people, places, and things; (3) **Geonames.** A geographical database. The different LOD datasets are interlinked via owl:sameAs statements which state that some individuals (represented via different URIs) denote the same object in the world.

Proceedings of the Joint Workshop on NLP&LOD and SWAIE, pages 16–22,
Hissar, Bulgaria, 12 September 2013.

The ontologies in LOD define the conceptualization of the world and provide mechanism for inference, search and maintenance over the data within the datasets. Thus ontologies are a powerful mechanism for representation and manipulation of knowledge. They can be used to define different views over the same information and facilitate inference for consistency checking. Unfortunately, the conceptualization encoded in ontologies could follow different ontological assumptions. Thus, when someone uses datasets from LOD, he/she has to pay attention to the way in which they are conceptualized. One solution for the diversity of the ontologies within LOD is the definition of a common unifying ontology.

Different communities have already defined and published their datasets on the LOD cloud. In the last years the linguistic community also moved towards the creation of Linguistic Linked Open Data (LLOD). The area has been under active development. Many of the community activities are reported here:

- The Open Linguistics Working Group: http://linguistics.okfn.org
- W3C Ontology-Lexica Community Group: http://www.w3.org/community/ontolex

(Chiarcos et al., 2012) list the following advantages:

- Representation and modeling
- Structural interoperability
- Federation
- Ecosystem
- Expressivity
- Conceptual interoperability
- Dynamic import

In this paper we rely on several of these advantages. More specifically, on the representation and modeling of language resources and their dynamic nature.

As an example of a knowledge-based language resource (KBLR) we follow the methodology for dynamic treebanking as it was implemented within the Redwood, DeepBank (Flickinger et al., 2012), BulTreeBank (Simov et al., 2004) treebanks and the infrastructure for dynamic treebanking INESS (Rosén et al., 2012). For example, the Redwoods treebank was compiled by coupling English Resource Grammar (ERG) and a tree selection module of [incr tsdb()] (see (Oepen, 1999) and (Oepen et al., 2002)). ERG produces very detailed syntacto-semantic analyses of the input sentence. For many sentences, HPSG processor (e.g. LKB — (Copestake, 2002)) overgenerates, producing analyses that are not acceptable. From the complete analyses different components can be extracted in order to highlight different views over the analyses: (1) derivation trees composed of identifiers of lexical items and constructions used to build the analysis; (2) phrase structure trees; and (3) underspecified MRS representations. From these types of information the most important with respect to the treebank construction is the first one, because it is good enough to support the reconstruction of the HPSG analysis by a parser. The steps of constructing the Redwood treebank are:

- LKB produces all possible analyses according to the current version of ERG;
- The tree comparison module provides a mechanism for selection of the correct analyses;
- The selection is done via basic properties (called also discriminating properties) which discriminate between the different analyses;
- The set of the selected basic properties are stored in the treebank database for later use in case of a treebank update.
- The update of the grammar initiates an update of the treebank itself. All the sentences that were annotated on the basis of the previous version of the grammar are analyzed again. The selection is done on the basis of the previous annotator selections.

We aim at designing and implementing a system that supports the creation and maintenance of language resources following the ideas of dynamic treebanking.

3 Knowledge-Based Language Resources

Our goal is to implement a system for supporting the whole life cycle of the creation and usage of

language resources at knowledge level. This process includes formal modeling of different aspects of the LRs management, such as:

- Representation of the annotation schema (e.g. a grammar). Here we envisage a representation of linguistic ontology, defining the vocabulary for describing linguistic objects and their basic properties, and constraints determining the actual linguistic objects.

- Representation of the annotation (analyses with respect to the annotation schema). We expect the annotation to be a representation of linguistic objects. Additionally, we require the implementation of appropriate tools for supporting creation of the annotation (automatically and/or manually). Checking consistency of annotation. Searching in the knowledge base.

- Creation (management) of language resources. Here we expect all the metainformation to be modeled and stored in the process of creation and evolution of LRs. Also, some of the activities in the process of creation and management of LRs to be supported on knowledge level.

- Usage of language resources. Originally each LR is created with a task in mind, but frequently this LR is used for many other tasks. In these cases usually the language resource is adapted to the new task including reformulation of the knowledge, stored in the LR, extensions with information from other resources and human intervention. Thus LR are directly connected to the actual usage. We view the knowledge-based approach to language resources management as being ideal to support all these requirements.

Modeling the annotation schema and annotations using ontologies is motivated by the need of complex and detailed language resources. Such language resources cannot be theory independent because of the following reasons:

- For real applications - detailed analyses are needed. The corresponding language resources need to incorporate the same level of granularity as the analyses necessary for the applications.

- On a certain level of granularity the annotation scheme becomes very complicated to be processed manually in a consistent way. Thus, a certain formalization will be necessary.

- On a certain level of granularity some linguistic theory has to be exploited. When it is possible, it is better to select existing theories instead of inventing a new "annotation" theory.

The example presented in the next section is based on the Head-driven Phrase Structure Grammar (HPSG). There are several reasons for this choice. HPSG is already formalized and this formalization allows a conversion between different formalisms. There exist examples of dynamic language resources created within this linguistic theory. The created framework could be easily ported to the setup of the knowledge-based language resources.

4 Dynamic Treebanking

This section is heavily based on (Simov, 2003). As it was stated above, we aim at modeling the HPSG dynamic treebanking as a knowledge-based language resource. First, we need to formalize the HPSG Language Model. It includes the following components:

- Linguistic objects in HPSG are represented as directed graphs called feature structures. The analyses of sentences are represented as complete feature structures in which all the constraints stated in the sort hierarchy and the grammar are satisfied.

- Sort hierarchy[1] defines a linguistic ontology. It represents the types of linguistic objects and their characteristics (features). The sort hierarchy determines the possible linguistic objects.

- Grammar is represented as a theory within a logical formalism over the sort hierarchy. It constraints the possible linguistic objects to the actual linguistic objects. The grammar is divided in two parts: (1) HPSG Universal and Language[2] Specific Principles; and (2) Language Specific Lexicon. Each principle and

[1] Sometimes it is called type hierarchy.

[2] English, Bulgarian, Tagalog or another natural language

each lexical entry is represented as a formula in the logical formalism. The formulas include sort assignment, equality of compositions of features as elementary formulas and full logical set of connectives over the elementary formulas.[3]

In order to support the HPSG dynamic treebank as knowledge-based language resources management we have to be able to support at least the following tasks:

- Implementation of an Annotation Schema. The sort hierarchy can be represented as an ontology in a straightforward way. The representation of formulas from the grammar is possible, but the standard inference for OWL DL can not be used, because OWL DL does not support equality.

- Analyses of sentences are represented as a set of complete instances. Without appropriate inference over the annotation schema we rely on an external processor which produces all the acceptable analyses over the grammar. These analyses are translated as RDF graphs using the ontology behind the annotation schema.

- Minimisation of manual work for the creation of the treebank. We select the correct analysis from the set of all analyses via classification of elementary formulas with respect to these analyses. After finite number of steps the correct analysis is selected. The process is described below.

- Adaptation of the treebank to different uses. Some usages require different views over the data. This is implemented by defining a new ontology for each new view over the treebank. See below.

- Dynamic nature is supported by a re-design of the annotation schema and re-classification of the existing analyses with respect to the new annotation schema. See below.

Corpus Annotation. As it is stated above, the corpus annotation within this framework is based on parse selection from a number of automatically constructed sentence parses. There are tree steps in the annotation process: (1) Pre-processing of the selected sentence. This step includes a segmentation of the text in sentences. Each sentence is annotated morphologically. (2) The result is encoded as a partial feature graph and it is further processed by an HPSG processor to a set of complete feature graphs. (3) The parse selection is considered as a classification with respect to the result from the previous step based on partial descriptions provided by the annotator.

Classification is done by means of index over the set of the produced parses. First, the intersection of all graphs is calculated. We assume that this part is true. Then an index is created on the basis of the elementary formulas within the graph. The index is a decision tree over the analyses. Each node in the tree is marked with an elementary formula which can be true or false. After selection of the correct value the corresponding edge is traversed to the next node. In this way the number of possible choices is reduced. The leaves of the tree are marked with the parses. Thus, in the index the formulas are chosen in such a way that each path from the root of the tree to some of its leaves determines exactly one graph in the initial set of graphs.

In the annotation the annotator supplies partial formulas (elementary formulas) about the true analysis. The index is traversed and the number of the possible choices is reduced. The process is repeated to the moment when only one analysis is selected. This analysis is stored within the treebank.

There is a problem with this annotation procedure. Any index represents just one view on the classification over the set of graphs. The annotator can have different views of classification over the same set of graphs. This means that at some moment the annotator could be not able to provide a formula that is in the index.

This clash between a predetermined way of classification scheme and the linguistic intuition could be resolved by construction by all the indices. In this case, the annotator chooses the most appropriate for him/her type of classification. These indices are represented as a forest in order to minimize the size of the representation. The annotation process proceeds according to the following classification algorithm:

1. The nodes in the index are available to be

[3] We will not be more specific here about the logical language and the interpretation of the language. Interested reader can consult the appropriate literature.

chosen by the annotator

2. The annotator decides on an elementary formula about the sentence
3. The elementary formula is found in the index and the number of the possible graphs is reduced
4. If there is only one possible graph, it is returned as a result, otherwise the algorithm returns to step 2

The number of the selections are in the worst case equal to the number of all analyses for the sentence. This can happen when the annotator rules out exactly one analysis per choice. The average number of selections is a logarithm from the number of the analyses. An important advantage of this selection-analysis-approach is that the annotator works locally. Thus, the number of parameters necessary to be considered simultaneously is minimized.

Corpus Update. If at some step of annotation the annotator cannot select any elementary formula from the index, we assume that there is no correct analysis in the set for the sentence. This fact is reported back to the grammar writer in order to modify the grammar. In such cases of grammar update the annotated sentences in the treebank needs to be reclassified with respect to the new annotation schema. The change of the annotation schema could be necessary in many cases like:

- Modifications in the target linguistic model of the elements in the corpus,
- New developments in the linguistic theory,
- Misleading decisions, taken during the design phase of the corpus development,
- New applications, for which the corpus might be adjusted.

In each of these cases the result is a new annotation schema. The treebank created with respect to the previous annotation schema could not be valid anymore with respect to the new one. The main question in this case is: How to use the existing corpus in the new circumstances and at minimal costs? In the case of knowledge-based language resources the idea of reclassification could be implemented as a transfer of all the relevant knowledge represented from the old annotation scheme to the new one. The transfer of the linguistic knowledge is defined by correspondence rules of the following format $\delta_{old} \Rightarrow \delta_{new}$, where δ_{old} is a formula with respect to the old annotation schema and δ_{new} is a corresponding formula with respect to the new annotation schema. In most cases the transfer rules are for formulas that are the same with respect to both annotation schemas. In such settings the reclassification algorithm for an HPSG treebank is as follows:

- For a sentence in the treebank we construct a new set of complete graphs $\{G_1, \ldots, G_n\}$,
- From the existing annotation we construct a set $EDnew$ of formulas using the correspondence rules,
- The set $EDnew$ is used for a classification of the sentences with respect to the set $\{G_1, \ldots, G_n\}$.

There could be the case when the transferred knowledge is not enough to classify the sentence with respect to the new set of graphs. In such a case manual intervention of an annotator will be necessary in order to complete the annotation.

Corpus Usage. Reclassification can be exploited in cases when the treebank will be used for a particular task which requires a different view over the represented linguistic knowledge. For example, for evaluation of parsers usually one needs to convert the treebank into a format comparable to the format of the parses. In case of knowledge-based language resources such evaluation could be done via representation of the parses with respect to annotation schema with the KBLR system. Then we apply appropriate reclassification of both: the treebank and the parses. Defining different annotation schema for the evaluation schema allows us to measure the performance of the parser with respect to different phenomena when this is necessary. For example, most of the parsers will be good on non-recursive phrases, but they will make mistakes over phenomena like PP attachment, coordination, etc. Thus, we could design annotation schema that hides the easy cases and compare the parses and the treebank on the hard problems.

Documentation of Annotation Process. Formalization of the annotation process via ontologies and knowledge bases of instances provide a rich mechanism for storing important information

from the process of the annotation. For this we envisage the usage of the so called annotation context ontology. This ontology describes the annotation setup: annotation schema, version of the HPSG grammar, annotators, time information, different annotation tasks. Each time some annotation action is performed appropriate record is created and stored. This will allow very detailed description of the annotation process. For example, it will allow to search for all the sentences annotated by some annotator, even after several modifications of the annotation schema and the eventual reclassification of the data.

Also such recording will help further usage of the treebank. For example, each usage could be reported with respect to the annotation context ontology. Later when somebody wants to use the treebank for evaluation of a new parser he/she could first examine the setups in which the treebank was used for similar tasks and then design a new one. Similarly the history of the annotation process could be useful in the process of design and implementation of new language resources.

Created in such a way language resources provide an easy connection to linked open data. Each knowledge-based language resource can be seen as a LOD dataset in which:

- The annotation schema is represented as an ontology,
- The annotated corpus is a set of instance data,
- The actual representation of the ontology and instance data in RDF is a trivial task,
- The documentation can be generated from the documentation of the annotation schema and published as a set of web pages dynamically,
- The description of the annotation process is also part of the dataset.

Integration with LOD is possible in both directions: (1) Any version of the created language resource could be immediately made available as a LOD dataset. This would facilitate the further use of the resource and feedback response as early as possible. (2) In some cases access the annotation process could gain from the access to other knowledge-based language resources. This can be done via inference mechanisms like classification and reclassification described above.

5 Conclusion and Future Work

This paper presented the main requirements for the creation of knowledge-based language resources on the basis of HPSG treebanking. As it is presented here, the actual implementation depends heavily on external processors like LKB system for producing HPSG analyses of sentences. In future, a system supporting knowledge-based language resources management will need integration with many external systems. But in our view, for such a system to be widely used, it needs to provide also internal tools working directly with ontologies and instance data. These tools need to support the complete manual annotation cycle, not just the selection of correct analysis. The cycle would include also: creation of processing procedure directly over RDF graphs like regular grammars, rules, transformation scripts, etc. These services will include many standard tools that already exist in the world of Linked Open Data. But there are also needs for new specific tools which are specially designed for the creation and management of KBLR. Another group of tools necessary to be implemented are: visualization and editing facilities.

In many respects we will follow the design and the implementation of CLaRK system — (Simov et al., 2003). In this context we consider RDF graphs corresponding to XML documents, ontologies corresponding to DTDs or XML schemas, etc. We believe that this is the way in which exploitation of language resources and technologies will be made widely used. Such a system is especially important within initiatives like CLARIN[4] whose huge target group of end users would like to exploit the available language resources and tools for their specific tasks. Many of them are not familiar with the principles behind the language resources and technologies. Knowledge-based system are perfect to support such kind of users. Especially important for them is the annotation context ontology which will provide them with access to the best practises in usage of language technologies. We envisage developing our system in this direction within the Bulgarian part of CLARIN infrastructure. We envisage extension of the annotation context ontology to incorporate also information about creation of language technologies and their dependency on language resources.

[4] www.clarin.eu

Acknowledgments

The work reported here is partially supported by the FP7 Capacity Project AComIn: Advanced Computing for Innovation (316087). I would like to thank Petya Osenova for many discussions on the topics reported in the paper and also for comments on early version of the paper. I am also grateful to colleagues from Ontotext AD who I have been working with on some aspects of LOD in the last years.

References

Christian Bizer, Tom Heath, and Tim Berners-Lee. 2009. Linked data-the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22.

Christian Chiarcos, John McCrae, Philipp Cimiano, and Christiane Fellbaum, 2012. *Towards open data for linguistics: Lexical Linked Data.*

Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars*. CSLI lecture notes: Center for the Study of Language and Information. CTR FOR STUDY OF LANG & INFO.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*, pages 85–96. Edições Colibri.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The lingo redwoods treebank motivation and preliminary applications. In *Proceedings of the 19th international conference on Computational linguistics - Volume 2*, COLING '02, pages 1–5, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stephan Oepen. 1999. *[incr tsdb()] - Competence and Performance Laboratory.* Saarland University.

Victoria Rosén, Paul Meurer, Gyri Smrdal Losnegaard, Gunn Inger Lyse, Koenraad De Smedt, Martha Thunes, and Helge Dyvik. 2012. An integrated web-based treebank annotation system. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*, pages 157–167. Edições Colibri.

Kiril Ivanov Simov, Alexander Simov, Milen Kouylekov, Krasimira Ivanova, Ilko Grigorov, and Hristo Ganev. 2003. Development of corpora within the CLaRK system: The BulTreeBank project experience. In *Proceedings of the 10th conference of the European chapter of the Association for Computational Linguistics*, EACL '03, pages 243–246, Budapest, Hungary.

Kiril Simov, Petya Osenova, Alexander Simov, and Milen Kouylekov. 2004. Design and implementation of the Bulgarian HPSG-based treebank. In *Journal of Research on Language and Computation, Special Issue*, pages 495–522, Kluwer Academic Publishers.

Kiril Ivanov Simov. 2003. Hpsg-based annotation scheme for corpora development and parsing evaluation. In Nicolas Nicolov, Kalina Bontcheva, Galia Angelova, and Ruslan Mitkov, editors, *RANLP*, volume 260 of *Current Issues in Linguistic Theory (CILT)*, pages 327–336. John Benjamins, Amsterdam/Philadelphia.